Onderzoeksrapport Nr. 8210

AN ALGORITHM FOR THE OPTIMAL CONTRACTION OF TOP-DOWN READABLE DECISION TABLES WITH GIVEN CONDITION ORDER

by

J. VANTHIENEN

Wettelijk Depot : D/1982/2376/15

Katholieke Universiteit Leuven, Department of Applied Economic Sciences.

This work was supported by the C.I.M., project no. 10.

CONTENTS

0. INTRODUCTION

1. PROBLEM STATEMENT
   1.1. General outline of the contraction process
   1.2. The top-down requirement
   1.3. The optimality criterion
   1.4. Relationship to other algorithms
   1.5. Multi-state conditions
   1.6. Dealing with impossibilities

2. THE CONTRACTION ALGORITHM
   2.1. Overview of requirements and/or limitations
   2.2. Used notation
   2.3. High-level description of the algorithm
   2.4. Refinements to the high-level algorithm
   2.5. Acceleration of the algorithm
   2.6. Illustrative example
   2.7. Execution performance

3. CONCLUSION

4. ACKNOWLEDGEMENTS

0. INTRODUCTION

Computer based procedural decision modeling aims at producing optimal decision table representations of new or existing procedures, regulations, etc. One of the optimality criteria is the number of columns in the final contracted table.

As the contraction process implies merging several decision situations leading to the same action configuration into one single condition combination, a minimal table length improves both the efficiency of automatic decision making as the clarity and ease of use by a human decision maker.

In this paper an algorithm is presented which minimizes the number of columns of the resulting decision table, for a given condition order and within the constraints stated below. It has been applied successfully in the interactive procedural decision modeling system, PRODEMO (1).

The PRODEMO system is a computer program for constructing and subsequently using decision tables. Its main purpose is to guide and support the user during decision modeling as well as during decision making by giving feedback and suggestions, by checking for incompleteness and inconsistencies and by executing all of the administrative routine drawings. Decision tables are put forward as a basic technique enabling the user to structure and check his procedural decisions, where the use of the interactive PRODEMO system largely enhances the capabilities of this technique by its editing, modeling, error checking, optimisation and decision making function.

---

(1) For general information on PRODEMO, see : MAES R., VANTHIENEN J. : PRODEMO : PROcedural DEcision MOdeling through the use of decision tables, Version 2, Bedrijfseconomische Verhandeling nr. 8101, K.U. Leuven, Department of Applied Economics, February 1981, 36 pp.

1. PROBLEM STATEMENT

1.1. General outline of the contraction process

A decision table describes the relationship between combinations of condition states and a number of corresponding actions. In certain situations however, different condition combinations lead to the same configuration of actions to be executed. One (or more) condition(s) might be irrelevant then, which means that the state of that condition does not influence the selection of the set of actions to be executed. When a condition is irrelevant with regard to a specific decision situation, it is indicated with a '-' (don't care entry) in the given decision table column.

A decision table with only one state for every condition in the decision columns is called an expanded table and the columns are therefore called expanded columns. A contracted table, on the other hand, contains columns with one or more don't care entries, also called contracted columns. (1)
E.g. :

columns

Y		N	
Y	N	Y	N
X	X	-	X
-	-	X	X

can be contracted into columns

Y	N	
-	Y	N
X	-	X
-	X	X

Fig. 1.

Contraction is not restricted to adjacent decision rules, but also applies to groups of columns, leading to the same action configuration and only differing in one condition. This leads to a don't care entry followed by one or more non-don't care entries.

---

(1) VERHELST, M. : De Praktijk van Beslissingstabellen, Kluwer-Deventer/ Antwerpen, 1980, 175 pp., p. 11.

E.g. :

Y		N	
Y	N	Y	N
X	-	X	-
-	X	-	X

leads to

-	
Y	N
X	-
-	X

Fig. 2.

The contraction process is reversible, which means that a table with don't-care entries can be transformed into a table with only expanded columns. This is called the expansion of a column or table.

The contraction process described in this paper, starts from an expanded table. It is however possible to construct a contracted table from the so called 'decision grid chart' without having to provide for a completely expanded table. This also offers the advantage that the information which is already incorporated in the decision grid chart can be used in order to reduce the contracting effort. We will deal with these items in 1.4.

1.2. The top-down requirement

It should be noted here that decision tables are both for manual and automatic use. Manual use requires that a table can easily be read from top to bottom, according to the notion of stepwise refinement. When filling in the states of the relevant conditions, the user should be able to gradually decrease the relevant part of the table on a straightforward basis until one single column is reached, or more concretely : every condition state answer corresponds to only one block in the condition part to be refined further.

E.g.

c1	Y		N	
c2	Y	N	Y	N
a1	X	-	-	-
a2	-	-	X	-

should NOT be replaced by

c1	Y	N	-
c2	Y	Y	N
a1	X	-	-
a2	-	X	-

Fig. 3 (a)

but (accidentally !) one could easily reorder the conditions in this case :

(b)

c2	Y		N
c1	Y	N	-
a1	X	-	-
a2	-	X	-

This last example shows the interaction between the contraction process and the condition order. The search for an optimal condition order (with a minimum number of columns), given the requirement of top-down refinement, constitutes a tough, time-and space-consuming problem (cfr. 1.3). The presented algorithm will therefore only produce a table of minimal size for a given condition order.

1.3. The optimality criterion

Optimality is defined here in terms of minimizing the number of columns and not in terms of execution efficiency of the resulting table, because

a) the tables are mainly constructed for manual use and the conversion or execution efficiency are subordinate to the clarity of the decision table which is improved by minimizing the number of columns (within the top-down readability constraint).

b) reducing the number of columns usually assists in improving the execution time
c) information on condition test times (t_i) and column frequencies (f_j) is hard to obtain when manual use is important
d) improved clarity enhances long term efficiency.

When optimality is translated into a minimal number of table columns, arrangements should be made for conditions having more than two states and for impossible condition combinations. These topics are considered in 1.5 and 1.6 respectively.

The algorithm only minimizes the table length for the given condition order, but this constraint is not as harmful as it might look, because
a) the condition order is often subject to precedence constraints or can even be fixed by the problem situation.
b) the original condition order often leads to an optimal solution (1) due to the experience or the feeling of the problem solver.

The search for an optimal order for n conditions (without repeating conditions), with the above restrictions (top-down readability), implies choosing between n! possible orders (2), as indicated in the following table :

---

(1) A sufficient (though not necessary) condition for optimality is when the number of columns in the contracted table equals the number of different action combinations, because columns with different action combinations are never contracted. Whenever this occurs, reordering conditions cannot improve the optimal solution anymore.
(2) This implies that every additional n-th condition multiplies the number of existing possibilities with n, as $n! = n \times (n-1)!$ and $0! = 1$.

no. of conditions	no. of possibilities
1	1
2	2
3	6
4	24
5	120
6	720
7	5040
8	40320
9	362880
10	3628800

It is obvious that an enumeration of all these possibilities is not recommended when 5 or more conditions are present. Other methods (e.g. branch and bound, dynamic programming ...) might lead to more interesting results here, but they are rather space-and time-consuming (1)

When the number of conditions is reasonably small, a simple (manual or automatic) enumeration leads to interesting results. This method becomes even more appealing when precedence constraints have to be considered (i.e. when the number of alternatives largely decreases). This function can be performed automatically in the PRODEMO system (2)

1.4. Relationship to other algorithms

The changing application field of the decision table technique from the programming area to the systems analysis and design phases and to the general procedural decisions area leads to the following important considerations :

---

(1) As no practical implementation of these methods is present in PRODEMO (up till now), they will not be considered here any further.

(2) All possible condition orders (within the precedence constraints) are enumerated and the order which results in the shortest or minimal table is adapted.

a) A lot of decision tables are not directly intended to be converted to computer programs. Conversion to optimal flow-charts or programs therefore is not always relevant.
b) The use of decision tables as a structuring tool in various procedural decision situations involves upgrading the decision representation capabilities above the real decision making act. When decision tables are used to improve the construction, flexibility, correctness and completeness of decision systems, they should be easy to use, top-down readable or shortly : structured.

Algorithms for the contraction of expanded decision tables (e.g. (1)) do not meet the top-down requirement (2) nor guarantee the absolute minimum number of columns. Though they apply to limited entry decision tables, they could easily be adapted to incorporate extended entries.

Algorithms for the immediate conversion of decision grid charts into compressed decision tables, avoiding the storage of the fully expanded decision table in memory (cfr. (3)) also lead to suboptimal tables which need to be contracted further. None of them produces top-down readable decision tables.

1.5. Multi-state conditions

As conditions can have more than two states, contraction should be able to cope with such conditions (4).

---

(1) SHWAYDER, K. : Combining Decision Rules in a Decision Table, Communications of the ACM, Volume 18, Number 8, August 1975, 476-480.
(2) The top-down requirement could easily be satisfied by repeating some of the conditions. This solution is not considered here.
(3) MAES, R., : An Algorithmic Approach to the Conversion of Decision Grid Charts into Compressed Decision Tables, Communications of the ACM, Volume 23, Number 5, May 1980, 286-293.
(4) A Multi-state condition is defined here as a condition with three or more possible mutually exclusive states and is also called an extended entry condition. A limited entry condition is a condition with two mutually exclusive states (usually YES and NO). VERHELST, M., op.cit., p. 82.

E.g.

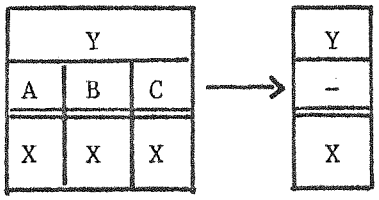

Fig. 4.

The same holds true for adjacent groups of columns, corresponding to three or more condition states.

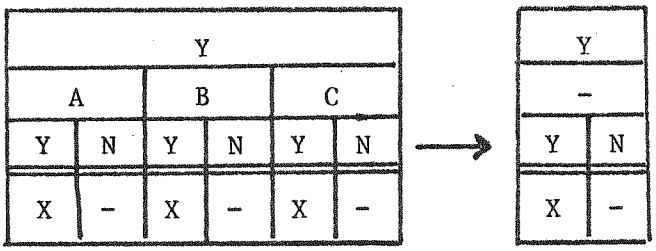

Fig. 5.

When conditions have more than two states, contraction is on an all/nothing basis, meaning that all states may be combined in a don't care entry ('-'), but not a subset of them. This is due to the lack of natural language understanding capability of present day computers. This need for semantic analysis is illustrated in the following examples :

E.g.

Radio	Color T.V.	B/W T.V.
X	-	-

should be contracted to

Radio	T.V.
X	-

Fig. 6.

and

$X \leq 1$	$1 < X \leq 2$	$X > 2$
X	X	-

$X \leq 2$	$X > 2$
X	-

Fig. 7.

Some arrangements could be made in these cases by some semi-automatic, interactive or partial procedures (1). Contraction of a subset of condition states is then no longer represented by a '-', but e.g. by an enumeration or concatination of the related states (2).

E.g.

color	Red	Green	Pink	Blue
action	X	X	-	-

→

color	Red,green	Pink,blue
action	X	-

Fig. 8.

1.6. Dealing with impossibilities

Impossibilities are condition combinations which are declared impossible, i.e. which can never occur, due to the nature of the problem.

E.g.

New customer	Y		N	
Age of account (years)	$\leqslant 1$	> 1	$\leqslant 1$	> 1
Rule no.	1	2	3	4

Fig. 9.

Rule no.2 is impossible in this case, because we are dealing with the account of a new customer. This implies that column 2 can be deleted from the decision table and the entry '$\leqslant 1$' in column 1 is called an implied

---

(1) We will not be dealing with these refinements as they have not been implemented yet in the PRODEMO system.

(2) A first improvement in this direction could be the combination of related states into an ELSE-state of the condition. Cfr. : MAES, R. : Bijdrage tot een kritische Herwaardering van de Beslissingstabellentechniek, K.U. Leuven, Doctoral Dissertation, 1981, 397 pp., p. 20. This is a non-optimal solution to the multi-state contraction process, as the ELSE is only able to deal with one subset of condition states.

condition entry. There are different ways to represent impossible condition combinations as indicated in fig. 10 : (1)

- assigning them to an additional action 'impossible'. (a)
- deleting the impossibility and indicating the implied condition entry with !, x or ( ). (b)
- simply deleting the impossible condition combination. (c)
- contracting possible and impossible entries into don't care entries. (d)

(a)

New customer	Y		N	
Age of account	⩽1	>1	⩽1	>1
Actions	X	-	X	X
Impossible	-	X	-	-

(b)

New customer	Y	N	
Age of account	⩽1 !	⩽1	>1
Actions	X	X	X

(c)

New customer	Y	N	
Age of account	⩽1	⩽1	>1
Actions	X	X	X

(d)

New customer	Y	N	
Age of account	-	⩽1	>1
Actions	X	X	X

Fig. 10.

---

(1) cfr. VERHELST, M., op.cit, p. 41.

If impossibilities are contracted (cfr. fig. 10.d), the resulting table will be shorter when adjacent groups can be combined. We will therefore choose this last representation for contracted tables and representation (c) for expanded tables (1), because the table is shorter and not loaded with information on (often evident) impossibilities.

Note : When some (not all) states of an extended entry condition are impossible in a certain condition combination, the impossible state(s) can not always be contracted with the possible state(s) and should be removed from the final table.

Because the computer can not contract (>1-≤10) and (>10) into (>1) : E.g.

Age of student	≤10		
Age of enrollment	≤1	>1 - ≤10	>10
action	1	2	Impossible

→

Age student	≤10	
enrollment	≤1	>1 - ≤10
action	1	2

Fig. 11.

End note.

2. THE CONTRACTION ALGORITHM

2.1. Overview of requirements and/or limitations

The contraction algorithm obeys the following requirements and limitations :
- execution speed of the algorithm should of course be very high, especially because of the interactive environment in which it has to be used.

---

(1) A disadvantage of this option is the difference in condition combinations between the expanded and the contracted table. Cfr. MAES R., op.cit.,p. 225.

- contraction should take into account the top-down readability constraint (cfr. 1.2.)
- the algorithm should contract (groups of) columns on an optimal basis (cfr. 1.3.).

Note : Optimality is defined here in terms of minimizing the number of columns and not in terms of execution efficiency of the resulting table, as the tables are mainly constructed for manual use.

- the condition order is fixed and given (cfr. 1.3.)
- the algorithm should be able to deal with multi-state conditions (cfr. 1.5.)
- impossibilities should be treated as indicated (cfr. 1.6.).

2.2. Used notation

The contraction algorithm is designed to operate on a matrix representation of the decision table, where conditions, condition states and actions are represented by their sequence numbers (1).

The following information is known :

knum : number of conditions

statnum (cond) : number of states per condition
with cond = 1, ..., knum

tlength : length (in columns) of expanded table $= \prod_{cond=1}^{knum} statnum(cond)$

tk(cond, column) : condition entry of condition in column (irrelevant = 0)
with column = 1, ..., tlength

anum : number of actions

aconfig (column) : action configuration in column
(i.e. a binary number containing a sequence of '1' and '0', according to 'X' and '-' in the action part of a column)

(1) For the conversion from this mathematical notation to the final decision table, see : VANTHIENEN, J. : A dynamic programming algorithm for displaying decision tables, K.U. Leuven, Onderzoeksrapport no.8006, 1980, 30pp.

impos (column) : indicates whether the column is possible (0) or impossible (1)

Additional variables are :

clength : number of columns needed for the (first) enumeration of all states of a condition within a subtable

$$\text{clength}(k) = \prod_{i=k}^{knum} \text{statnum}(i)$$

for limited entry tables : $\text{clength}(k) = 2^{knum-k+1}$

slength : number of columns needed for the enumeration of one state of a condition $\text{slength}(k) = \text{clength}(k)/\text{statnum}(k)$

$$= \prod_{i=k+1}^{knum} \text{statnum}(i)$$

for limited entry conditions : $\text{slength}(k) = 2^{knum-k}$

statbegin : column where the first state of a condition (re)starts. The distance between these columns is the value of clength.

start 1 : column where a state of a condition (re)starts. The distance between these columns is the value of slength.

displacement, col 1, col 2, lastpos, i, j : column indicators for the comparison of the states starting in start 1 and start 1 + slength

col 1 = start 1 + displacement

col 2 = col 1 + slength

lastpos = last possible column.

Note : clength and slength are not defined as arrays in the algorithm, because their values for a specific condition can easily be computed from their values for the previous condition (i.e. a division by statnum (k)).

End note.

2.3. High level description of the algorithm

The algorithm presented below transforms the expanded decision table into a contracted table, according to the following procedure :

> For all conditions, the successive occurrences of their states are examined. If the set of action configurations is equal for all states, the entries are replaced by '-' and the duplicate columns are marked to be deleted.

This procedure implies in fact a subdivision of the original table according to the condition values from the first to the last condition, as illustrated by the tree structure in fig. 12.

When all the subtables which originate from the same node, are equal, the corresponding condition entry is irrelevant. Equal subtables in fig. 12 have been indicated.

Note : Before subdividing the various (sub)tables, one could first check whether all action combinations within the (sub)table are equal. If they are, all lower conditions become irrelevant. This process is repeated starting from the completely expanded table to the subtables obtained after subdivision of knum-1 (thereby contracting the last condition).
This technique, however, did not add to the efficiency of the algorithm, due to technical reasons (checking all lower condition entries, marking the columns for deletion, etc...).

End note.

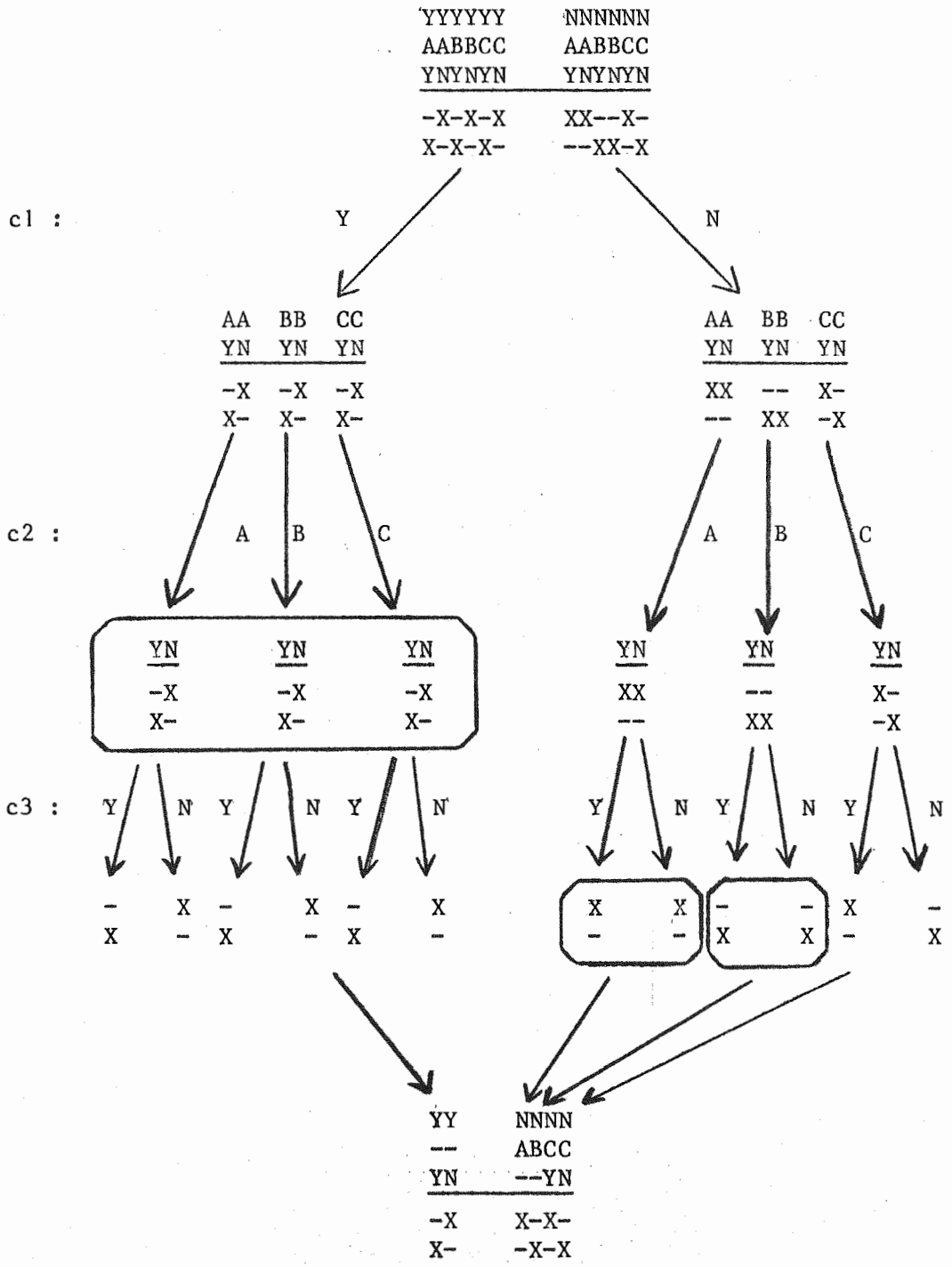
YYYYYY NNNNNN
AABBCC AABBCC
YNYNYN YNYNYN
-X-X-X XX--X-
X-X-X- --XX-X
c1 :
Y
N
AA BB CC
YN YN YN
-X -X -X
X- X- X-
AA BB CC
YN YN YN
XX -- X-
-- XX -X
c2 :
A
B
C
YN YN YN
-X -X -X
X- X- X-
YN YN YN
XX -- X-
-- XX -X
c3 :
Y N Y N Y N
- X - X - X
X - X - X -
Y N Y N Y N
X X - - X -
- - X X - X
YY NNNN
-- ABCC
YN --YN
-X X-X-
X- -X-X

Fig. 12.

Contraction occurs when the subtables are equal, i.e. when the corresponding columns of each subtable refer to the same action configuration. Impossibilities are considered equal to every other action configuration, as we took the option to contract them with possible columns (cfr. 1.6.). If impossibilities are treated like an additional action or explicitly entered this way (cfr. 1.6.), the impossible columns can easily be contracted with other impossible columns.

The contraction algorithm uses the information which is reflected in the number of states per condition, in order to compute the location of the various subtables. Contraction therefore replaces the condition entries with '-', but does not remove the duplicate columns from the expanded table. These columns are simply marked for deletion and deleted at the end.

As the remaining impossible columns (which have not been contracted) also have to be deleted after the contraction (cfr. 1.6.), the duplicate columns are considered as impossibilities and are therefore also marked with : impos(column) = 1.

Bearing in mind that contraction proceeds from the first to the last condition and for each condition from left to right, the structure of the algorithm is described in fig. 13 and the high level form is as given in fig. 14.

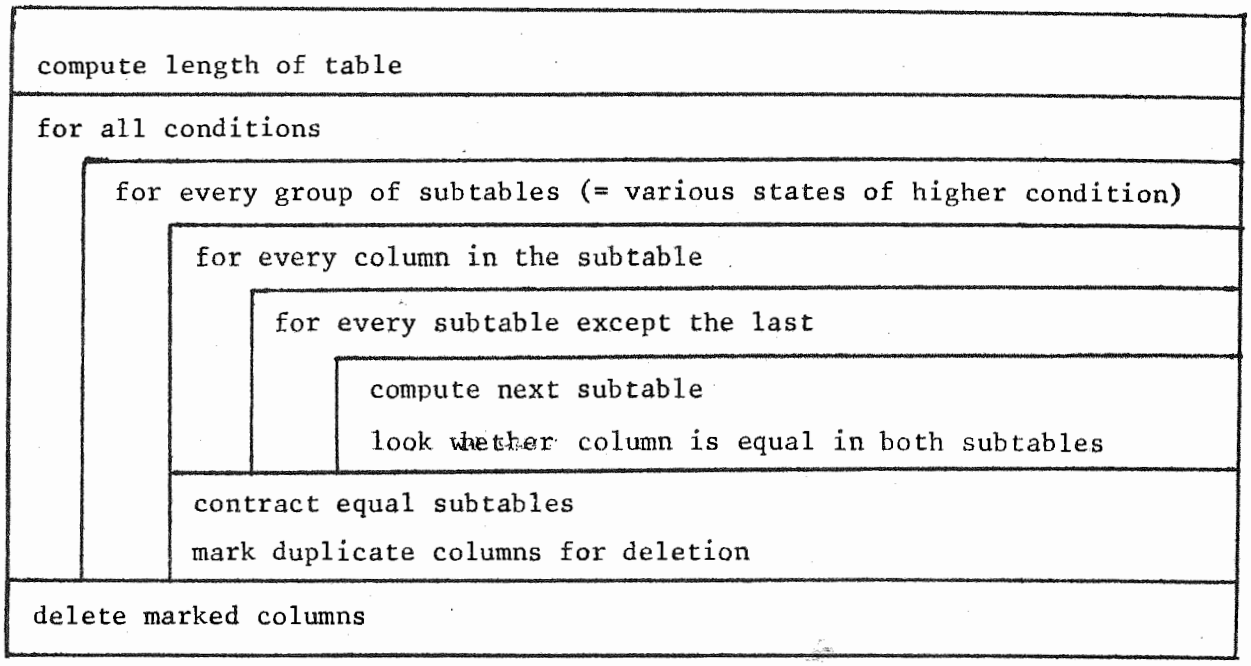

Fig. 13.

```
compute tlength
clength : = tlength
for cond : = 1, knum, 1
    slength : = clength / statnum (cond)
    for group statbegin : = 1, tlength, clength
        for column displacement : = 0, slength - 1, 1
            for subtable start 1 : = statbegin, statbegin + clength -
                                                    2*slength, slength
                col 1 : = start 1 + displacement
                col 2 : = col 1 + slength
                LOOK WHETHER COL1 AND COL2 ARE EQUAL
            end for
        end for
        CONTRACT THE SUBTABLES IF EQUAL
    end for
    clength : = clength / statnum (cond)
end for
DELETE MARKED COLUMNS
```

Fig. 14.

2.4. Refinements to the high-level algorithm

a. Some refinements still have to be made to the above high level algorithm, i.e. first : LOOK WHETHER COL1 AND COL2 ARE EQUAL.

As already mentioned, possible columns are equal if they have the same action configuration. Two impossible columns are also considered equal. If only one of the columns is possible, it should then be equal to a previous corresponding column (if any). There are two cases :

- only the first is possible : this column has already been compared with a previous possible column (if any)
- only the second is possible : this column needs to be compared with the last possible column (if any).

We therefore need to keep the location of the last possible column (in =lastpos=).

If two possible columns are found with unequal action configurations, the subtables cannot be contracted and the algorithm continues with the next group of subtables.

These refinements for the comparison between col1 and col2 are given in the following decision table (fig. 15).

Note : By checking all subtables for a given column, the need for a multi-dimensional variable = lastpos = is avoided. This is the reason for the order of the two inner for-loops in fig.13 and fig.14.

End note.

b. Another refinement to the algorithm of fig. 14 is the contraction of (groups of) columns which were found equal, i.e. : CONTRACT THE SUBTABLES IF EQUAL. The subtables are found equal if both inner for-loops have not been interrupted, i.e. when displacement = slength-1.

LOOK WHETHER COL1 AND COL2 ARE EQUAL							
col 1 possible ?	Y			N			
col 2 possible ?	Y		N	Y			N
lastpos already found ?	-		-	Y		N	-
(col 1, col 2) same configuration ?	Y	N	-	-		-	-
(lastpos, col 2) same configuration ?	-	-	-	Y	N	-	-
lastpos : = col 1	X	X	X	-	-	-	-
go on to next group :	-	X	-	-	X	-	-
displacement:=slength							
start 1:=statbegin+clength-2*(slength)+1							
columns are considered equal	X	-	X	X	-	X	X

Fig. 15.

The contraction then replaces the entries of the corresponding condition by '-' in the first subtable and deletes (marks) the other subtables. A column in the first subtable however is only possible if it was possible in any of the subtables.

This refinement is given in fig. 16.

c. The last refinement to be made concerns the deletion of the marked columns in the contracted table, i.e. DELETE MARKED COLUMNS.
These columns (if any) have been contracted or found impossible and should be deleted in order to obtain the final decision table. The other columns should fill the gaps which originate from this deletion.

This is obtained by defining a variable (=last column=) indicating the last remaining column, cfr. fig. 17.

Notice that the value of tlength is no longer the length of the expanded table, but the value of =last column=, i.e. the length of the contracted decision table.

```
CONTRACT THE SUBTABLES IF EQUAL
if displacement = slength-1
   then for i : = statbegin, statbegin+slength-1,1
           tk(cond,i): = 0
           for j : = i, i+ clength-slength+1, slength
              if impos(j) = 0
                 then impos(i) : = 0
                      aconfig(i): = aconfig(j)
                      j : = i + clength-slength+1
              end if
           end for
        end for
        for i : = statbegin + slength, statbegin+clength-1,1
           impos(i): = 1
        end for
        comment : there are marked columns now to be deleted
end if
```

Fig. 16.

```
DELETE MARKED COLUMNS
lastcolumn : = 0
for column : = 1, tlength,1
   if impos(column) = 0
      then lastcolumn : = lastcolumn+1
           if column ≠ lastcolumn
              then condition configuration (lastcolumn):=condition confi-
                                                    guration (column)
              aconfig (lastcolumn):=aconfig (column)
              impos (lastcolumn): = 0
           end if
   end if
end for
tlength : = lastcolumn
```

Fig. 17.

2.5. Acceleration of the algorithm.

When, for a given condition, columns have been contracted, the duplicate columns cannot be deleted, but are simply marked. The following condition should not attempt to contract these marked columns as they will be removed from the final table, so contracting them is a waste of time.
The algorithm should proceed as follows (fig. 18) :

E.g.
```
YYYYYYNNNNNN
YYYNNNYYYNNN
ABCABCABCABC
------------
123123444444
```

contraction of condition 1 : - no changes -

contraction of condition 2 :
```
YYYYYYNNNNNN
---NNN---NNN
ABCABCABCABC
------------
123123444444
  ___   ___
 marked  marked
```

contraction of condition 3 :
```
YYYYYY   NNNNNN
---NNN   ---NNN
ABCABC   -BCABC
---------------
123123   444444
  ___      ____
 marked   marked
```

When contracting condition 3, columns (4,5,6) and (10,11,12) should not be considered anymore

Fig. 18.

In order to accelerate the algorithm, the marked columns are not considered anymore, i.e.one immediately looks for the next possible column and calculates the position where the corresponding subtable starts.

The high level form of the algorithm (fig. 14) then looks as follows :

```
compute tlength
clength : = tlength
for cond : = 1,knum,1
    slength : = clength / statnum(cond)
    i:=1
    while i ⩽ tlength
        if impos(i) = 1
            then i:=i+1
            else statbegin := integer [(i-1)/clength] * clength+1
                 start2 : = integer [(i-1)/slength] * slength+1
                 for column displacement :=0, slength-1,1
                     for subtable start 1:=start2,statbegin+clength-2*slength,
                                                                    slength
                         col1 : = start 1 + displacement
                         col2 : = col1 + slength
                         LOOK WHETHER COL1 AND COL2 ARE EQUAL (Fig. 15)
                     end for
                 end for
                 CONTRACT THE SUBTABLES IF EQUAL (fig. 16)
                 i:= statbegin + clength
        end if
    end while
    clength : = clength / statnum(cond)
end for

DELETE MARKED COLUMNS (fig. 17)
```

Fig. 19.

2.6. Illustrative example

As an illustration of the contraction process, consider the following expanded table :

Book	hard cover				normal edition				pocket			
Customer	wholesaler		retailer		wholesaler		retailer		wholesaler		retailer	
Quantity	⩽100	> 100	⩽100	> 100	⩽100	> 100	⩽100	> 100	⩽100	> 100	⩽100	> 100
Discount	5%	10%	5%	5%	0%	5%	0%	5%	0%	0%	0%	0%

Fig. 20.

In mathematical notation, this table is given by :

1	11	11	22	22	33	33
2	11	22	11	22	11	22
3	12	12	12	12	12	12
1	23	22	12	12	11	11
column	12	34	56	78	910	1112

Fig. 21.

The contraction process will now be illustrated for every condition :

- condition 1 : there are 3 subtables (1-4, 5-8, 9-12). As the 3 action configurations are not equal, condition 1 cannot be contracted.
- condition 2 : subtables (1-2, 3-4) : different action combinations
  subtables (5-6, 7-8) : equal action combinations, so condition 2 is irrelevant in column 5-6, and columns 7-8 are marked for deletion.

subtables (9-10,11-12) : equal action combinations, so condition 2 is irrelevant in column 9-10 and columns 11-12 are marked for deletion.

The table now looks as follows :

1	1111	2222	3333
2	1122	0022	0022
3	1212	1212	1212
1	2322	1212	1111
column	1234	5678	9101112
marked		78	1112

Fig. 22.

- condition 3 : subtables (1,2) : different action combinations

subtables (3,4) : equal action combinations, columns 3-4 contracted and column 4 marked for deletion

subtables (5,6) : different action combinations

subtables (7,8) : already marked

subtables (9,10): equal action combinations, columns 9-10 contracted and column 10 marked for deletion

subtables (11,12):already marked

The table is now contracted and looks as follows :

1	1111	2222	3333
2	1122	0022	0022
3	1202	1212	0212
1	2322	1212	1111
column	1234	5678	9101112
marked	4	78	101112

Fig. 23.

After deletion of the marked columns :

1	111	22	3
2	112	00	0
3	120	12	0
1	232	12	1

Fig. 24.

Replacing the numbers by their original names results in :

Book	hard cover			normal edition		pocket
Customer	wholesaler		retailer	-		-
Quantity	$\leqslant 100$	> 100	-	$\leqslant 100$	> 100	-
Discount	5%	10%	5%	0%	5%	0%

Fig. 25.

2.7. Execution performance

As the contraction algorithm has to be used in an interactive environment, the execution speed should be very high.

The time needed to perform the contraction process depends on :

- the number of conditions
- the number of expanded table columns
- the number of contractions to be done (1)

(1) Subtables can only be contracted if all action configurations have been found equal. If, however, a different action configuration is encountered, the comparison stops and restarts with the next group of subtables, without having to analyze the other columns of the subtables.

As far as PRODEMO is concerned, processing time for a complete contraction of a table with maximal dimensions is 0.353 seconds. Contraction of tables with normal dimensions takes on average 0.022 seconds. (1)(2).

Total elapsed time when executing a contraction (i.e. the time between the contraction request and the end of the contraction, as experienced by the user) shows an average of 2-3 seconds and a maximum of 5 seconds for large table dimensions (1). This time includes user feedback and time-slice interrupts.

These results are within all reasonable limits in order to be used in an interactive system.

3. CONCLUSION

The contraction algorithm described in this paper, is able to contract decision tables on an optimal basis with regard to the number of table columns and within the requirement of top-down refinement.

It was designed to deal with extended entry conditions and with impossible condition combinations.
Both the very low storage requirements and the high performance of the presented algorithm enabled a very useful incorporation into the PRODEMO system.

The algorithm however is only able to minimize the table length for a given condition order. Further research is needed in order to deal with condition order optimization.

---

(1) Lowest priority figures on the CDC PLATO CYBER computer
(2) A PRODEMO table reaches its maximal dimensions at 240 distinct decision columns.

4. ACKNOWLEDGEMENTS

The algorithm and its description in this paper benefited from the valuable cooperation with Prof. Dr. M. VERHELST and Prof. Dr. R. MAES. The author wishes to thank them for all comments and suggestions.